POLITECHNIKA GDAŃSKA

Język Java podstawy

Jacek Rumiński

Język Java podstawy

Jacek Rumiński

Katedra Inżynierii Biomedycznej,
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

1. Wprowadzenie do modelowania obiektowego

2. Klasy i konstruktory

3. Klasy i dziedziczenie

Ale – Co to jest „klasa”

Język Java jest językiem obiektowym. Oznacza to, że tworzenie programów związane jest z pewnym sposobem modelowania tego, co nas otacza oraz tego, co sobie wyobrażamy. Programowanie obiektowe wymusza zastosowanie określonej formy kodu źródłowego oraz umożliwia wykorzystanie szeregu ciekawych cech, które poznamy później.

Klasa oznacza pewien typ (rodzaj, kategorię).

Posłużmy się przykładem ryb. Już samo pojęcie „ryba” oznacza pewien „rodzaj” (pojęcie ogólne). Czy istnieje ryba jako ryba? Nie. Istnieje konkretna ryba, którą złapaliśmy na wędkę : ma określone cechy: kolor łusek, ilość płetw oraz określone zachowanie (np. wije się – węgorz?). Co więcej, złapana ryba istnieje konkretnie (w danym miejscu i czasie – na haczyku, teraz).

Zatem: RYBA = KLASA (zestaw cech i zachowania jakie mają ryby)
ZŁAPANA RYBA = OBIEKT klasy RYBA (konkretne wystąpienie typu).

Ale – co to jest „klasa”

Opiszmy rybę:

klasa RYBA:	obiekt klasy RYBA
cechy:	
kolor oczu;	=niebieskie
kolor łusek;	=pomarańczowe
liczba płetw,	=5
płeć;	=samica
czy_drapieżna;	=nie
zachowanie:	
pływa;	pływa
żeruje;	żeruje

Ale – co to jest „obiekt”?

Jeśli utworzyliśmy (zaprojektowaliśmy) daną klasę (RYBA) to możemy utworzyć wiele obiektów danej klasy z różnymi cechami.

OBIEKT – wystąpienie (instancja) KLASY

Obiekty w świecie rzeczywistym istnieją konkretnie (w czasie i miejscu). Obiekty w świecie komputerowym będą „żyły” również w określonym czasie i miejscu - w pamięci (operacyjnej RAM, lub utrwalone w pamięci nieulotnej).

```
okoń1 = nowy obiekt (klasy RYBA);
okoń1(kolor oczu)=czerwony;
okoń1(płeć)=samiec;
…
```

```
okoń2 = nowy obiekt (klasy RYBA);
okoń2(kolor oczu)=czerwony;
okoń2(płeć)=samica;
…
```

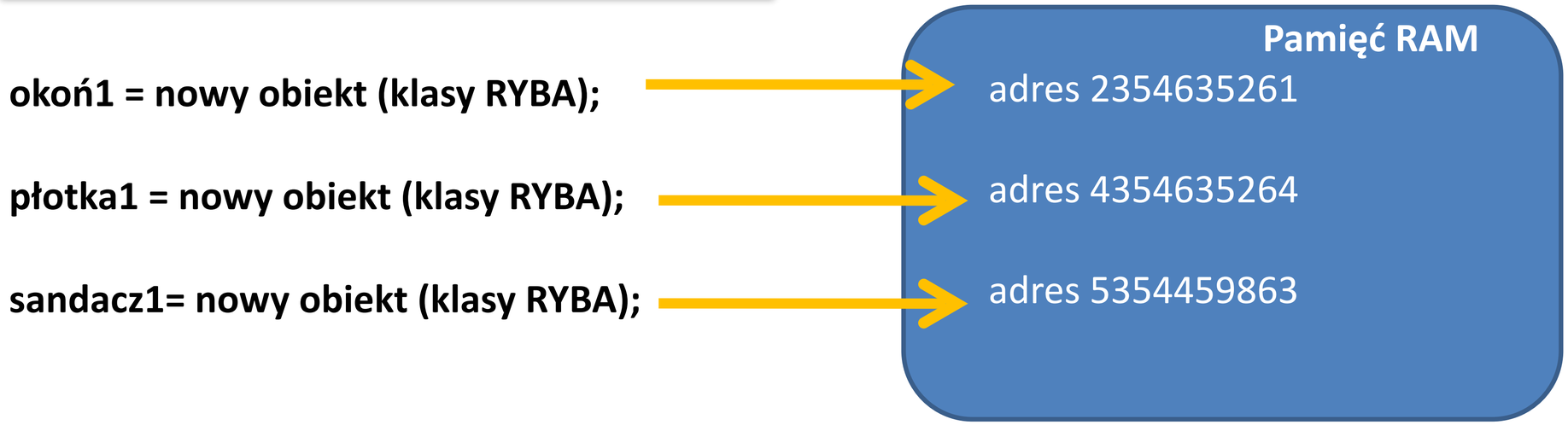
Możemy utworzyć wiele obiektów danej klasy !!!

Ale – co to jest „obiekt”?

```
okoń1 = nowy obiekt (klasy RYBA);
okoń1(kolor oczu)=czerwony;
okoń1(płeć)=samiec;
...
```

Konstruktor – „tworzy” obiekt

Ustawienie wartości cech danej (konkretnej) ryby



Nazwy: „okoń1”, „sandacz1” to „uchwyty” wskazujące na pamięć (o jaki obiekt nam chodzi).

Skrót myślowy: obiekt „Okoń1” – znaczenie: obiekt pod adresem...

Ale – co to jest „obiekt”?

Tożsamość i unikalność obiektu określana jest nie przez wartości cech ale przez instancję (tj. konkretne wystąpienie w określonym czasie pod danym adresem pamięci).

Dwa obiekty mogą mieć TE SAME wartości wszystkich cech, ale są różne (NOT EQUAL). Mają inną lokalizację w czasoprzestrzeni (przestrzeń - pamięć RAM).

Modelowanie obiektowe posiada szereg innych cech (np. dziedziczenie klas – typ KOBIETA jest podtypem – dziedziczy po typie CZŁOWIEK), które omówimy później.

Teraz musimy przedstawić zasady zapisu klas i obiektów w języku JAVA.

Każdy język obiektowy będzie miał własną metodę (składnię) zapisu definiującego klasy i obiekty. Komputery są bardzo konkretne – konkretny (i dokładny co do znaku) musi być również zapis kodu źródłowego definiujący klasę i obiekt.

Jak zamodelować klasę w JAVIE?

Każdy język programowania ma zestaw słów (zwykle w j. angielskim) zarezerwowanych do zapisu treści (instrukcji, rozkazów) programu.

Do zamodelowania pierwszej klasy wykorzystamy następujące słowa kluczowe:

class - początek definiowania klasy,
public - specyfikator oznaczający dostępność (public – wszystkie obiekty mogą korzystać),
int - typ danych liczb całkowitych (integer),
void – typ danych oznaczający „nic”,
boolean – logiczny typ danych (wartość true lub false),
String – typ danych znakowych (który też jest klasą, stąd pisany z wielkiej litery).
Ponadto zastosujemy oznaczenie nawiasów klamrowych do określania (grupowania) bloku kodu – { }

Kod programu: Ryba*.java*

```
public class Ryba {
        //cechy - wartości mogą być unikalne dla każdego obiektu - nie ma „static”
        String kolor_lusek;
        int liczba_pletw;
        boolean drapiezna;

        //konstruktor
        public Ryba (String kl, int lp, boolean d){
                kolor_lusek=kl;
                liczba_pletw=lp;
                drapiezna=d;
        }//koniec Ryba()

        //działanie – funkcje, metody
        public void plywa(){
                System.out.println("Tak plywam: plum plum plum! ");
        }//koniec plywa()

}//koniec klasy Ryba
```

Jak zamodelować klasę w JAVIE?

W przykładzie konstruktor to funkcja trzech zmiennych. Można go wywołać podstawiając wartości danego typu – tworzymy obiekt z określonymi wartościami cech.

Zamiast używać różnych nazw zmiennych (np. nazwa pola: „kolor_lusek”, nazwa zmiennej „kl”) można wykorzystać kolejne słowo kluczowe this („w tym obiekcie”).

```
String kolor_lusek;
int liczba_pletw;
boolean drapiezna;
//konstruktor
public Ryba (String kolor_lusek, int liczba_pletw, boolean drapiezna){
        this.kolor_lusek=kolor_lusek;
        this.liczba_pletw=liczba_pletw;
        this.drapiezna=drapiezna;
}//koniec Ryba()
```

Czy to był program?

1. Kompilacja Ryba.java - OK.

2. Uruchomienie – NIE!!!

Co utworzyliśmy? Klasę, z której możemy skorzystać tak samo jak z wielu klas dostępnych w standardowej dystrybucji Javy.

Klasę można traktować jako najmniejszy element biblioteki, z której możemy wielokrotnie korzystać (ang. reuse).

Ale jak coś uruchomić!?

Już wiemy - musimy użyć funkcji „main()”.

Jak korzystać z klas?

W czasie ostatniego wykładu podaliśmy sobie przykład klasy, opisującej (modelującej) rybę.

Dzisiaj rozpatrzmy bardzo podobny przykład klasy modelującej Rycerza Jedi !

źródło: www.play.com

Kod programu: RycerzJedi.java

```
public class RycerzJedi{

        //pola  - zmienne obiektu
        String nazwa;
        String kolor_miecza;

        //konstruktor
        RycerzJedi(String nazwa, String kolor_miecza){
                this.nazwa=nazwa;
                this.kolor_miecza=kolor_miecza;
        }

        //metody – funkcje obiektu
        void opis(){
                System.out.println("Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz.");
        }

}// koniec class RycerzJedi
```

A co z wykorzystaniem klas?

Dwa warianty:

1. Wszystko w jednym pliku (nie za dobrze – wersje rozwojowe). Tylko jedna klasa może być wówczas publiczna (oznaczona jako public) – i jest to główna klasa aplikacji (musi zawierać funkcję main()). Nazwa pliku – taka sama jak nazwa klasy publicznej !

2. Każda klasa w swoim pliku (tak lepiej – można wielokrotnie używać i rozwijać daną klasę).

PRZYKŁADY

Kod programu: ZlotJedi.java

```
class RycerzJedi{
        String nazwa;
        String kolor_miecza;
        RycerzJedi(String nazwa, String kolor_miecza){
                this.nazwa=nazwa;
                this.kolor_miecza=kolor_miecza;
        }
        void opis(){
                System.out.println("Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz.");
        }
}// koniec class RycerzJedi
 public class ZlotJedi{
        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])
}// koniec public class  ZlotJedi
```

Kod programu: RycerzJedi.java

```
//TAK JAK WCZEŚNIEJ
```

Kod programu: ZjazdJedi.java //w tym samym katalogu co RycerzJedi.java

```
public class ZjazdJedi{

        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])

}// koniec public class  ZjazdJedi
```

Tak dużo lepiej – teraz wiele klas może korzystać z klasy RycerzJedi !!!

Zmienne static i nie static
wyjaśnienie przez przykład - demonstracja

Jacek Rumiński

Pakiety – co to?

Klasy mogą być definiowane w ramach pakietu. Jak dotąd tego nie robiliśmy, stąd pakiet był domyślny.

Nazwy pakietów wprowadza się po to, aby grupować (logicznie, tematycznie, organizacyjnie) klasy. Pakiety mogą zawierać pakiety, itd.
Przykładowo standardowe klasy Javy umieszczono w pakiecie o nazwie „java”. Podstawowe elementy języka w pakiecie „java.lang”, klasy związane z obsługą we/wy w pakiecie „java.io”, itd.

Pakiet (zbiór pakietów/klas) przechowywany jest jako katalog. Każdy zawarty pakiet to podkatalog. Każda klasa to plik NazwaKlasy.class (w kodzie źródłowym to NazwaKlasy.java).

Pakiety są wygodne – ale nieobowiązkowe !

Pakiety – co to?

Nazwy pakietów to często odwrócona nazwa dziedzinowa (adresów WWW). Przykładowo moje pakiety i klasy mógłbym przechowywać w pakiecie: pl.gda.biomed.jwr
Jak definiować pakiet – słowo kluczowe package a potem nazwa pakietu (wszystko w pierwszej linii kodu). Przykładowo:

```
package pl.gda.biomed.jwr;
```

Jak wskazywać pakiety, z których klas chcę skorzystać:
Słowo kluczowe import i nazwa pakietu wraz z klasą lub nazwa pakietu z *. Przykładowo:

```
import java.io.*;  //wskazuję kompilatorowi, że będę korzystał z wielu klas
import java.net.Socket; //wskazuję, że będę używał klasy Socket z pakietu
```

Przejdźmy do przykładu ->

Kod programu: RycerzJedi.java

```
package biomed.jwr.knights;
public class RycerzJedi{

        // Zawartość jak poprzednio, tylko pola, konstruktor i funkcja oznaczone jako public

}// koniec class RycerzJedi
```

Kod programu: ZebranieJedi.java

```
package biomed.jwr;
import biomed.jwr.knights.RycerzJedi; //lub import biomed.jwr.knights*;
public class ZebranieJedi{
        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])
}// koniec public class  ZebranieJedi
```

1. Wprowadzenie do modelowania obiektowego

2. Klasy i konstruktory

3. Klasy i dziedziczenie

Klasy i obiekty

Klasa – typ obiektów,
Obiekt – instancja, wystąpienie, realizacja klasy.

Jak stworzyć klasę?

Obserwujemy rzeczywiste obiekty i zbieramy informacje o ich cechach oraz zachowaniu.
Jeśli współdzielą grupę cech i funkcji to jest to ich wspólny typ (kategoria, forma, itd.).
Ilość cech i rodzaj funkcji jakie zdefiniujemy jest taka, jaka jest potrzebna ze względu na cel tworzenia oprogramowania.

Przykładowo – liczba włosów na głowie jest cechą klasy Człowiek, ale czy w praktyce jest to po coś potrzebne zbierając informacje o osobach?

Konstruktor klasy

➢ Konstruktor klasy to specjalna metoda, która zwraca referencję do obiektu danej klasy. Oznacza to, że nie deklaruje się typu danych wartości zwracanej.

➢ Nazwa konstruktora musi być taka sama jak nazwa klasy (zgodność wielkości liter !!!).

➢ Dla jednej klasy można zdefiniować wiele konstruktorów, każdy tak samo się nazywa, lecz musi mieć różną liczbę argumentów lub różne typy argumentów.

Nazwa metody wraz z liczą argumentów i typami danych argumentów określana jest sygnaturą metody (unikalność!).

Konstruktor klasy

Konstruktor klasy może wywołać konstruktor klasy nadrzędnej (w przypadku gdy dana klasa dziedziczy po innej) oraz może wywołać inny konstruktor tej samej klasy.
Kolejność wołania konstruktorów w kodzie danego konstruktora jest następująca:

```
NazwaKlasy(argumety){
 this(argumenty1); //wywołanie innego konstruktora tej samej klasy
 super(argumenty1); //wywołanie konstruktora klasy bazowej
 kod;
}
```

Kod programu: KonstruktoryJedi.java

```
public class KonstruktoryJedi{

        int typ;
        /**wykonaj konstruktor bez argumentów, domyślne wartości*/
        public KonstruktoryJedi(){
                this(1); //wykonaj konstruktor z 1 argumentem
        }
        /**wykonaj konstruktor z 1 argumentem*/
        public KonstruktoryJedi(int typ){
                this.typ=typ;
        }
        //Wykonać raz z jednym, raz z drugim konstruktorem
        public static void main(String a[]){
                KonstruktoryJedi kj = new KonstruktoryJedi();
                //KonstruktoryJedi kj = new KonstruktoryJedi(50);
                System.out.println("Numer typu Jedi to: "+kj.typ);
        }//koniec main()

}//koniec public class KonstruktoryJedi
```

Kod programu: SuperKonstruktoryJedi.java

```
class WzorzecJedi{
        int typ;
        public WzorzecJedi(){       this(1);        }
        public WzorzecJedi(int typ){   this.typ=typ;          }
}
public class SuperKonstruktoryJedi extends WzorzecJedi{
        public SuperKonstruktoryJedi(){
                //domyślnie wykonaj super() -> WzorzecJedi()
        }
        public SuperKonstruktoryJedi(int typ){
                super(typ); //wykonaj -> WzorzecJedi(typ)
        }
        public static void main(String a[]){
                SuperKonstruktoryJedi kj = new SuperKonstruktoryJedi();
                //SuperKonstruktoryJedi kj = new SuperKonstruktoryJedi(50);
                System.out.println("Numer typu Jedi to: "+kj.typ);
        }//koniec main()
}//koniec public class SuperKonstruktoryJedi
```

Konstruktor klasy

Jeśli nie zdefiniujemy jawnie ŻADNEGO konstruktora, automatycznie tworzony jest domyślny, pusty (bez argumentów, bez instrukcji) konstruktor:

```
NazwaKlasy(){
}
```

Jeśli jednak napisany został jakikolwiek konstruktor – konstruktor domyślny nie będzie dostępny.
Zawsze możemy sprawdzić jakie są konstruktory w dokumentacji klasy.

Po utworzeniu obiektu, mając referencję do obiektu możemy sprawdzić jakiej klasy jest dany obiekt (operator instanceof):

```
luke instanceof Rycerz
```

1. Wprowadzenie do modelowania obiektowego

2. Klasy i konstruktory

3. Klasy i dziedziczenie

Zapraszamy na kolejne zajęcia